# CSE 523 | End term Report
# Building a natural language dialog system app for monitoring caloric intake

Prasoon Rai (110921754) Devesh Sisodia (110951296)

## 1 Introduction

"A Natural Dialogue System is a form of dialogue system that tries to improve usability and user satisfaction by imitating human behaviour" [1] (Berg, 2014). It addresses the features of a human-to-human dialog (e.g. sub dialogues and topic changes) and aims to integrate them into dialog systems for human-machine interaction.

In this project, we aim to build command line interface (CLI) that implements a natural language dialog system, which, given a description of food intake instance from an end user, reports the total caloric and related dietary information about the consumption. The system exercises a series of dialogues with the user in order to extract precise information about the food items consumed, before finalizing the best match with the database and reporting calorie values.

## 2 Spoken Dialog Systems

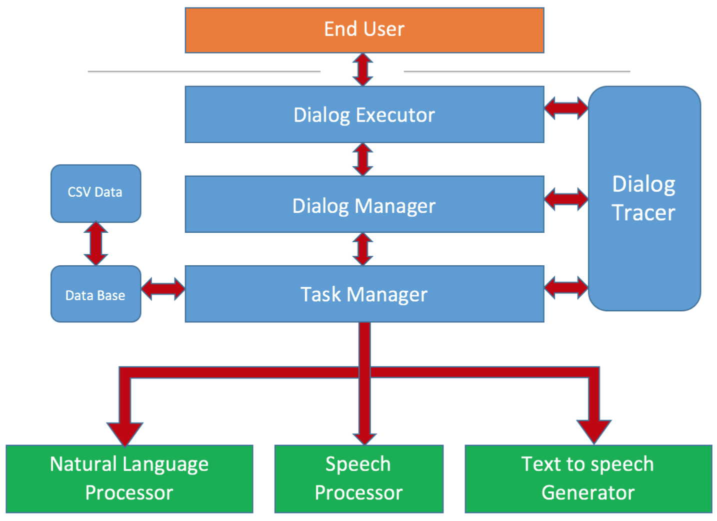
Before discussing our design, we briefly list the components involved typically in a speech based dialog system[2]:

- **Automatic Speech Recognizer (ASR):** The ASR decodes speech input from the user into text that acts as an input to the natural language understanding module in the system.
- **Natural language understanding:** It transforms a recognition into a concept structure that can drive system behavior. Thus, the translated input text from ASR is interpreted in context of the current application to extract requisite information.
- **Dialog manager:** The dialog manager controls turn-by-turn behavior. A simple dialog system may ask the user questions then act on the response. Such directed dialog systems use a tree-like structure for control.
- **Text-to-speech Generator (TTS):** It realizes an intended utterance as speech. Depending on the application, TTS may be based on concatenation of pre-recorded material produced by voice professionals, or a voice assistant like Google assistant, Apple Siri etc.
- **Response Generator:** It is similar to text-based natural language generation, but takes into account the needs of spoken communication. This might include the use of simpler grammatical constructions, managing the amount of information in any one output utterance and introducing prosodic markers to help the human participant absorb information more easily.
- **Domain Reasoner/Task Manager:** The domain reasoner, or more simply the back-end, makes use of a knowledge base to retrieve information and helps formulate system responses. In simple systems, this may be a database which is queried using information collected through the dialog.

We built our system to closely adhere to these design principles to ensure modular implementation and correctness. We describe our architecture next.

## 3 System Architecture

Following is the design layout of different components. A brief description of each module's functionality follows. In code, each of these correspond to a unique class with related methods, implemented in separate files with similar names.

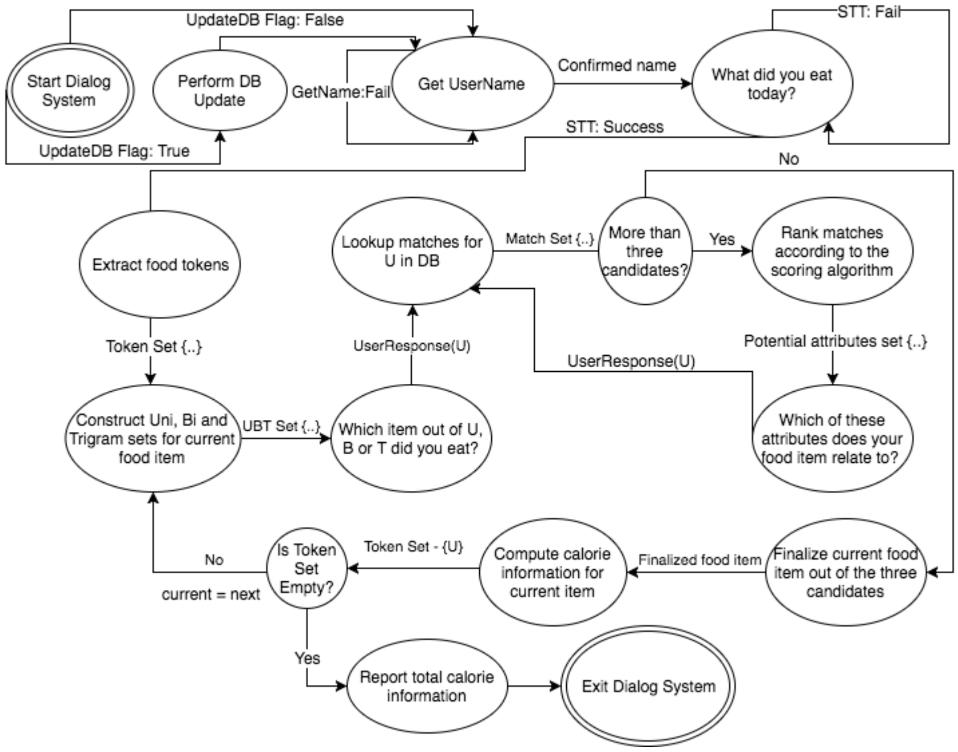


*Modules in our dialog system and their flow hierarchy*

- **End User:** The human subject who interacts with the dialog system. Its role is to provide food item consumption and responses to Dialog system's queries in order to assist in converging to a precise database entry.
- **Dialog Executor:** This module forms the application interface to the user and is responsible for exchanging information between the dialog manager and the user.
- **Dialog Manager:** All the dialog statements intended by the system towards the user are generated in this module. It also collects user responses and based on them, accordingly assigns tasks to the task manager.
- **Task Manager:** It is responsible for continuously collect tasks from the dialog manager, interpret them, assign to one of the task processors and finally hand over the processed output back to the dialog manager.
- **Natural Language Processor:** It handles all the tasks that require text processing for knowledge extraction, lexical analysis, text tokenization etc, to interpret user responses as system needed information.
- **Speech Processor:** It has methods for transforming user speech responses to text feedback and related operations.
- **Text to Speech(TTS) Generator:** This module converts text data from the dialog system into a voice response intended for the end user.
- **Dialog Tracer:** It implements the logger method to provide functionality for encapsulating system, dialog and user responses as dialog traces which are archived for studying system performance.
- **Data Base(DB):** This module has all the methods which are responsible for handling database query and management.
- **CSV Data:** It contains methods used for stream lining CSV data into composite data structures that can be queried for various information by the higher layers of abstraction in the design.

## 4 System State Machine and Implementation Flow

In this section we describe the state diagram of our system and run through a typical instance to understand information flow.


Start Dialog System
UpdateDB Flag: False
UpdateDB Flag: True
Perform DB Update
GetName:Fail
Get UserName
Confirmed name
What did you eat today?
STT: Fail
STT: Success
No
Extract food tokens
Token Set {..}
Lookup matches for U in DB
Match Set {..}
More than three candidates?
Yes
Rank matches according to the scoring algorithm
Potential attributes set {..}
Which of these attributes does your food item relate to?
UserResponse(U)
UserResponse(U)
Construct Uni, Bi and Trigram sets for current food item
UBT Set {..}
Which item out of U, B or T did you eat?
Finalize current food item out of the three candidates
Finalized food item
Compute calorie information for current item
Token Set - {U}
Is Token Set Empty?
No
current = next
Yes
Report total calorie information
Exit Dialog System

*High level system state machine illustration*

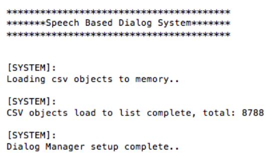
- **Start Dialog System:** This is the first invocation. As soon as the dialog system is initiated, DB update is performed if the DB_overhaul flag is set. If not, pickle dumps containing complete database information per object is loaded to memory.
- **Perform DB Update:** Updates DB update by invoking the crawler. This involves following steps:
  - Invoke database crawler to fetch csv links for all the food categories from the remote database[3].
  - Extract csv data into dictionary objects.
  - Persist dictionary objects in the memory as pickle dumps.
- **Get UserName:** Once we have all the content in memory, we begin user interaction. The first service in this flow is Get UserName which asks[4] user for their name, converts it to text using ASR[5], extracts name using NLP module and gets confirmation before moving on to the next task. Up to two iterations are done in case of misinterpretation, after which the user name is defaulted to *User*.
- **What did you eat today?:** In this step the end user is expected to provide a few sentence description of the food items that were consumed. One may refer to the example traces document to have a look at sample instances. Up to two retries are made in case of misinterpretation.
- **Extract food tokens:** Once the user response is received correctly, the ASR module translates it to text. The successfully parsed text is then passed to the NLP module to extract food tokens.
- **Construct Uni/Bi/Tri grams for food tokens:** The food token set is then discretized to form food item combinations as uni/bi/tri grams per food item.
- **Which item (U/B/T) did you eat?:** The dialog system then tries to figure out which of the U/B/T combination was user's consumed food item. Based on user response one is finalized.

- **Look up matches for U in DB:** After the current food item is finalized, its matches are looked up in the DB and a set of candidates is created. The next steps focus on reducing the candidate space to final three candidates.
- **More than 3 candidates?:** This decision making state is queried repeatedly until we have less than 4 candidate food matches for the current item.
- **Rank matches:** The potential candidates are ranked according to their probability of being user's food item. The top three candidates are created as the potential attribute set.
- **Which of these attributes relates?:** From the three candidates, the user can pick one as the best match, reject all the three, or simply say *not sure*. Based on this user response, the search space gets modified.
- **Finalize one of the three candidates:** Once we have the top three candidates, the user is asked to pick the best match. In this way, the current food item is matched down to one from the entire candidate set.
- **Compute calorie information:** Now that we have the food item finalized, the task manager computes the calorie information for the current item and stores it. The current food item is removed from the token set.
- **Is Token Set Empty?:** If we have exhausted all the food tokens that were extracted initially, we have the dietary information for the complete consumption. In that case, move to *report total calorie state*, otherwise proceed for the next food item by constructing U/B/T combinations and enter the dialog-feedback chain again.
- **Report total calorie information:** The calorie information for each food item collected in *Compute calorie information for current item* is parsed to display the detailed dietary information on the overall consumption data.
- **Exit Dialog System:** Since the user has been provided the requisite information, the dialog system is safely then exited.

## 5 Sample run and results

Below is the Sample Run of Our Project.

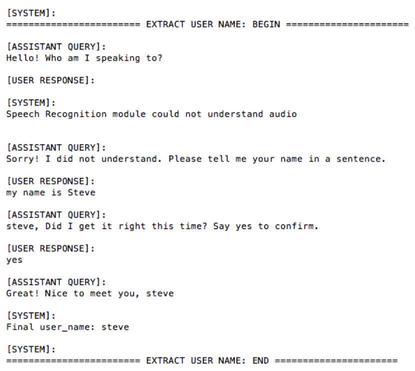
- **Dialogue System Intiation :** This Section Invokes th Database Class and Setup the Database, and load all the objects.



```
****************************************
*******Speech Based Dialog System*******
****************************************


[SYSTEM]:
Loading csv objects to memory..

[SYSTEM]:
CSV objects load to list complete, total: 8788

[SYSTEM]:
Dialog Manager setup complete..
```

*Dialogue System Initialization*

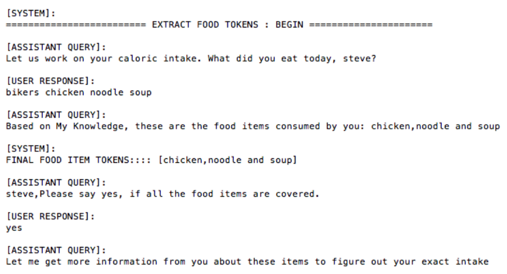
- **Getting User Name :** This Phase request user's name. And use this name to refer the user in the trailing conversation.



```
[SYSTEM]:
======================== EXTRACT USER NAME: BEGIN ======================

[ASSISTANT QUERY]:
Hello! Who am I speaking to?

[USER RESPONSE]:

[SYSTEM]:
Speech Recognition module could not understand audio


[ASSISTANT QUERY]:
Sorry! I did not understand. Please tell me your name in a sentence.

[USER RESPONSE]:
my name is Steve

[ASSISTANT QUERY]:
steve, Did I get it right this time? Say yes to confirm.

[USER RESPONSE]:
yes

[ASSISTANT QUERY]:
Great! Nice to meet you, steve

[SYSTEM]:
Final user_name: steve

[SYSTEM]:
======================== EXTRACT USER NAME: END ======================
```

*Extracting User Name*

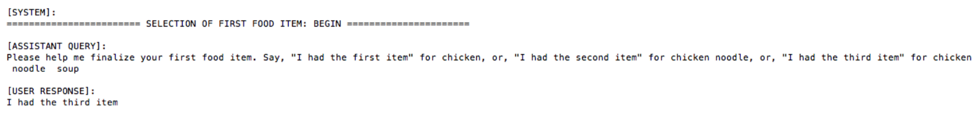
- **Extract Food Token :** This Phase asks for user what was his/her food Intake today. Based on the response food tokens are extracted based on our knowledge via USDA.



```
[SYSTEM]:
======================== EXTRACT FOOD TOKENS : BEGIN ======================

[ASSISTANT QUERY]:
Let us work on your caloric intake. What did you eat today, steve?

[USER RESPONSE]:
bikers chicken noodle soup

[ASSISTANT QUERY]:
Based on My Knowledge, these are the food items consumed by you: chicken,noodle and soup

[SYSTEM]:
FINAL FOOD ITEM TOKENS:::: [chicken,noodle and soup]

[ASSISTANT QUERY]:
steve,Please say yes, if all the food items are covered.

[USER RESPONSE]:
yes

[ASSISTANT QUERY]:
Let me get more information from you about these items to figure out your exact intake
```

*Capturing Food Tokens*

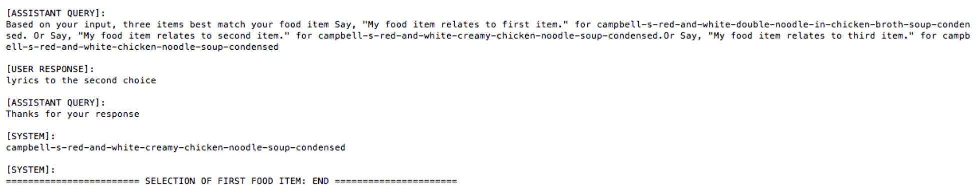
- **Capturing the food items sequentially :** This phase will work to capture the food item by selecting the food tokens corresponding to the food item.



```
[SYSTEM]:
======================== SELECTION OF FIRST FOOD ITEM: BEGIN ======================

[ASSISTANT QUERY]:
Please help me finalize your first food item. Say, "I had the first item" for chicken, or, "I had the second item" for chicken noodle, or, "I had the third item" for chicken
 noodle  soup

[USER RESPONSE]:
I had the third item
```

*Capturing Food Item*

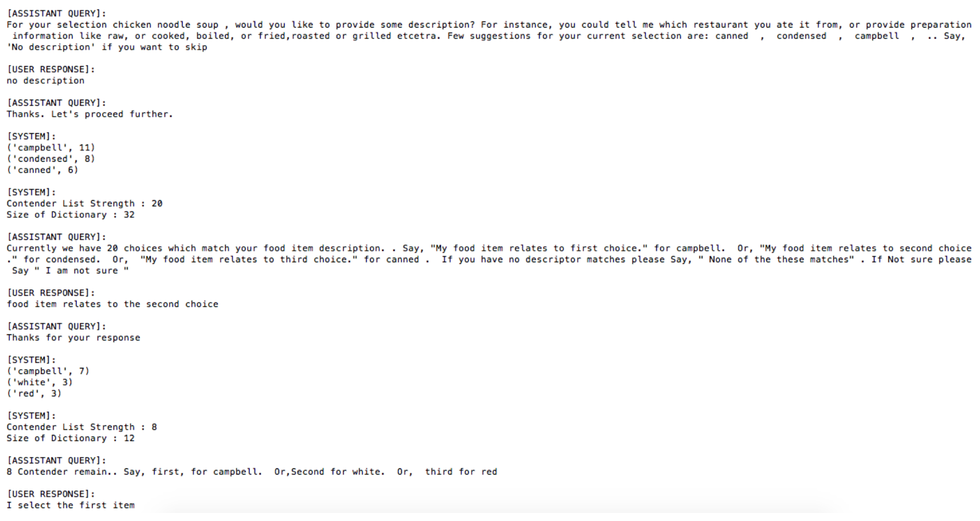
- **reducing the Contender List per food item :** This phase will find the set of contender which match the Database. We work to narrow the Contender in this phase recursively till we reach a stage where there are 3 or less contender remaining. We thus find the best match of the item in the USDA Database using user's feedback.



```
[ASSISTANT QUERY]:
Based on your input, three items best match your food item Say, "My food item relates to first item." for campbell-s-red-and-white-double-noodle-in-chicken-broth-soup-conden
sed. Or Say, "My food item relates to second item." for campbell-s-red-and-white-creamy-chicken-noodle-soup-condensed.Or Say, "My food item relates to third item." for campb
ell-s-red-and-white-chicken-noodle-soup-condensed

[USER RESPONSE]:
lyrics to the second choice

[ASSISTANT QUERY]:
Thanks for your response

[SYSTEM]:
campbell-s-red-and-white-creamy-chicken-noodle-soup-condensed

[SYSTEM]:
======================== SELECTION OF FIRST FOOD ITEM: END ======================
```

*Narrowing Contender List*

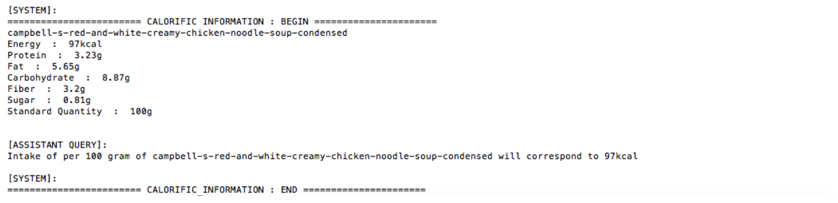
- **Selecting the Food Item :** Once the contender list is reduced to less than four, we throw those best three choices to the user in order to pick the best matching product.



```
[ASSISTANT QUERY]:
For your selection chicken noodle soup , would you like to provide some description? For instance, you could tell me which restaurant you ate it from, or provide preparation
 information like raw, or cooked, boiled, or fried,roasted or grilled etcetra. Few suggestions for your current selection are: canned  ,  condensed  ,  campbell  ,  .. Say,
'No description' if you want to skip

[USER RESPONSE]:
no description

[ASSISTANT QUERY]:
Thanks. Let's proceed further.

[SYSTEM]:
('campbell', 11)
('condensed', 8)
('canned', 6)

[SYSTEM]:
Contender List Strength : 20
Size of Dictionary : 32

[ASSISTANT QUERY]:
Currently we have 20 choices which match your food item description. . Say, "My food item relates to first choice." for campbell.  Or, "My food item relates to second choice
." for condensed.  Or,  "My food item relates to third choice." for canned .  If you have no descriptor matches please Say, " None of the these matches" . If Not sure please
 Say " I am not sure "

[USER RESPONSE]:
food item relates to the second choice

[ASSISTANT QUERY]:
Thanks for your response

[SYSTEM]:
('campbell', 7)
('white', 3)
('red', 3)

[SYSTEM]:
Contender List Strength : 8
Size of Dictionary : 12

[ASSISTANT QUERY]:
8 Contender remain.. Say, first, for campbell.  Or,Second for white.  Or,  third for red

[USER RESPONSE]:
I select the first item
```

*Finalizing Food Item*

- **Calorific Information :** This phase extract the calorific information of the food item. And displays the calorific information like Energy, Protein, Fat, Carbohydrate, Fiber, Sugar per 100 gm.



```
[SYSTEM]:
======================== CALORIFIC INFORMATION : BEGIN ======================
campbell-s-red-and-white-creamy-chicken-noodle-soup-condensed
Energy  :  97kcal
Protein  :  3.23g
Fat  :  5.65g
Carbohydrate  :  8.87g
Fiber  :  3.2g
Sugar  :  0.81g
Standard Quantity  :  100g


[ASSISTANT QUERY]:
Intake of per 100 gram of campbell-s-red-and-white-creamy-chicken-noodle-soup-condensed will correspond to 97kcal

[SYSTEM]:
======================== CALORIFIC_INFORMATION : END ======================
```

*Calorific Information*

## 6 Future Work

In this project, we were successfully able to demonstrate that an interactive dialog system can be created that takes uses voice as the only means of communication. We plan to undergo several enhancements to the system:

- **Decentralization:** This would be the first item on the upcoming features. The current implementation requires us to setup each machine with required libraries and database. We plan to host the application on cloud and provide the interface as a web application, accessible platform independently on mobile (all versions), desktop etc.

- **From CLI to UI:** The current application uses a command line interface. The cloud application will require a web based user interface for interaction.

- **User Profiles:** It is imperative to note that profiling users would help us in cataloging their information, study dietary patterns, *guess* their food items based on history and so on. We can also implement user authentication once we have this feature.

- **Improving potential candidate generation:** Currently we use frequency of matching attributes and present them to the user in descending order. We plan to optimize this to operate on earlier user feedback and profile to ask more relevant questions.

- **Align the application to *real* physician/patient interactions:** We plan to study the data provided by the university hospital archives of actual patients and doctors to modify the system to ask questions that are similar to these actual dialogues.

- **Integrate more databases:** We plan to utilize other food databases besides USDA to have better coverage on potential candidates, thereby increasing the accuracy of the model in terms of finalized item with respect to the actual consumed one.

- **Use pictures and other interactive cues:** Finally, we can use pictures, animations etc to assist the user in accurately determining the food items that were consumed in the respective proportions.

## 7 References

[1] Berg, Markus M. (2014), Modelling of Natural Dialogues in the Context of Speech-based Information and Control Systems, Akademische Verlagsgesellschaft AKA, ISBN 978-3-89838-508-4

[2] Spoken Dialog Systems wiki: https://en.wikipedia.org/wiki/Spoken$_{d}ialog_{s}ystems$

[3]$USDAdatabase : https : //ndb.nal.usda.gov/ndb/search/list$

[4]$GoogleTexttoSpeech(TTS)API : https : //techcrunch.com/2009/12/14/the - unofficial - google - text - to - speech - api/$

[5]$GoogleCloudSpeechtoTextAPI : https : //cloud.google.com/speech/$